Constrained Delegation

Olav Bandmann*
Swedish Institute of Computer Science (SICS)
Box 1263, SE-164 29 Kista, Sweden
olav@sics.se

Mads Dam†
LECS/IMIT, Royal Institute of Technology (KTH)
KTH Electrum 229, SE-164 40 Kista, Sweden
mfd@kth.se

Babak Sadighi Firozabadi*
Swedish Institute of Computer Science (SICS)
Box 1263, SE-164 29 Kista, Sweden
babak@sics.se

Abstract

Sometimes it is useful to be able to separate between the management of a set of resources, and the access to the resources themselves. Current accounts of delegation do not allow such distinctions to be easily made, however. We introduce a new model for delegation to address this issue. The approach is based on the idea of controlling the possible shapes of delegation chains. We use constraints to restrict the capabilities at each step of delegation. Constraints may reflect e.g. group memberships, timing constraints, or dependencies on external data. Regular expressions are used to describe chained constraints. We present a number of example delegation structures, based on a scenario of collaborating organisations.

1 Introduction

Consider the following *motivating example*: Organisation A produces some form of electronic documents which it regards as sensitive for some reason. The documents may have commercial value, or they may be classified in a military sense. Organisation A wishes to outsource some administrative task concerning its IT system to some other organisation, B. Included among B's tasks will be the assignment of access rights, according to policies established by A. For instance, a user or customer of A wishing to access some document should, if the request adheres to A's policies, be assigned that right by B. Not included among B's privileges, on the other hand, should be the right to access the documents for itself.

The natural solution to this problem is to use delegation. A wishes to delegate to B some administrative privilege over some resource, though not necessarily the privilege to use the resource for itself. In our approach we make this distinction explicit and we give a formalism for representing fine-grained delegation of privileges both of access-level and management-level type as explained in [5].

Acknowledging the danger of muddying further an already somewhat infected terminology of *delegation*, the purpose of this paper is to propose a new view of delegation, based on two key ideas:

1. The use of regular expressions to constrain the shape of delegation trees.
2. The capability of delegators - principals that issue delegations - to further refine those constraints as the delegation trees are being constructed.

By means of (1) we achieve enough expressiveness to easily handle our motivating scenario, as well as many more of a more realistic shape. By means of (2) we make sure that the expressiveness does not get out of hand — as few

*Supported by a project grant by Microsoft Research, Cambridge
†Supported by the Swedish Research Council, grant 281-98-653, "Semantics and Proofs for Programming Languages"

constraints as necessary need be given up front, and as the delegation tree is gradually built up, new constraints can be introduced as needed.

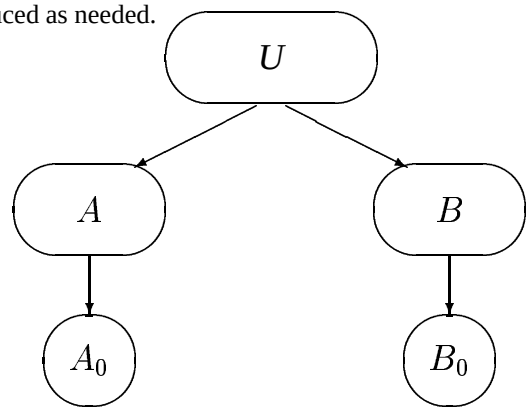

Figure 1. Example group hierarchy

To illustrate the approach let α be some authorisation, such as the right to read document x. Consider the group hierarchy shown in fig. 1. Here, U is some global group for the example, A and B are the groups (organisations) of the motivating example, A_0 is the owner of x, and B_0 will be the initial receiver in B of authorisation from A_0. Being the owner, A_0 is expected to possess all delegation rights concerning α. In our approach this is expressed by a certificate, or access control entry, of the shape

$$d_0 = (A_0U^*, \alpha, t_0)_S$$

where

- A_0U^* is a constraint expressing that A_0 is authorised to pass on α rights to U in zero or more steps,
- t_0 is time of issuance
- S is issuer (initially left unspecified)

Now, A_0 wishes to transfer to B_0, a specific subgroup of B trusted by A_0 for this purpose, the authority to create an organisation within B for assigning α privileges to members of A. This is achieved by A_0 issuing the following certificate:

$$d_1 = (B_0B^*A, \alpha, t_1)_{A_0}$$

This certificate is regarded as valid since:

1. It is issued by A_0.
2. d_0 assigns privileges concerning α to delegate to (refinements of) U^*
3. B_0B^*A is a refinement of U^* as a regular language, given the group hierarchy of fig. 1.

The certificate d_1 expresses an authorisation for B_0, namely the right to issue new certificates of the shape

$$d_2 = (A', \alpha, t_2)_{B_0}$$

where A' is some subgroup of A, or maybe of the shape

$$d_3 = (B_1B_2^*A, \alpha, t_3)_{B_0}$$

in this way step by step creating, within B, an organisation with authority to administer the rights (α) within A.

Observe that d_1 and its derivatives can only be used to grant α authorisations to members of A, so if we assume that A and B are disjoint, no member of B can use d_1 to grant α to itself.

The objective of this paper is to motivate and introduce this model of delegation, in the hope it will be seen as contributing a new and interesting mechanism for transferring authority between organisations in a flexible and controllable way.

Although this work is influenced by the work in the *Trust Management* area (see [2, 1, 8, 7] and [3]), its goal and focus is somewhat different. In this work, we do not address the issue of distribution of privileges as it is done in e.g. Simple Public Key Infrastructure (SPKI) [3]. In our model, we assume that there is a central authorisation server that verifies each delegation attempt separately. The focus of this work is instead on how to decentralise, in a controlled and verifiable way, the management (administration) of rights. The authorisation server as a central verifier will approve delegations as well as access permissions based on earlier approved delegations and certain global information such as revocations.

Earlier work on delegation has considered the virtues and otherwise in imposing controls on the shape of delegation trees. In [3], in particular, it is argued that, in SPKI, a principal possessing the right to delegate some permission should also have the right to delegate that permission to herself. This issue highlights an important way in which our setup differs from that of SPKI. In SPKI, authorisations are bound to public-private key pairs. A principal possessing a delegation right must also have the right to produce a key pair to which the delegated authorisation is bound. This key pair it can acquire for itself, of course.

So, if the application at hand requires distinctions to be made between permissions and the power to create permissions, the SPKI model of binding authorisations to key pairs must be somehow amended to allow key bindings to be constrained, or alternatively some other mechanism, such as ACL's, must be used.

In the paper we introduce and motivate the concept of constrained delegation. The paper focuses squarely on the handling of delegation trees; we are not concerned with issues of distribution, binding, or enforcement mechanisms.

We give, in section 2, a simple set-based semantical model, formalising the central notions of delegation chain, chain constraints, certificate, and authorisation. On this basis we establish, in section 3, a soundness result providing a basic healthiness property for the relationship between delegation chains and certificates. In section 4 we proceed to give a possible syntactical representation for chain constraints. The semantical framework imposes few restrictions on the way this is done. Here one proposal is given, based on a restricted form of regular expressions. We discuss some issues involved in choosing a good representation and give a couple of examples, mainly to illustrate the constructions that are involved. Then, in section 5, a more comprehensive scenario is discussed based on the idea of a number of national defence task forces delegating authority to a joint UN command. In a first reading of the paper it may be worthwhile to skip directly to this section, before going into the formal definitions. Several issues discussed briefly in the conclusion are left for future work, including revocation semantics, static and dynamic constraints, and practical realisations.

2 The Formal Model

2.1 Certificates

The fundamental notion is that of a *constraint*. In this paper the nature of constraints is left primitive. For all practical purposes it suffices to think of constraints as (time-varying) group membership constraints, as above. So constraints will be equipped with a partial order of entailment, or containment, and there will be a satisfaction relation explaining when (at what times) a constraint will be satisfied by a given principal.

Definition 1 (constraint structure) *Let $\mathcal{P}$ be a set, the set of* principals. *We denote the natural numbers by $\mathcal{T}$ (as in* time*). A* constraint structure *is a triple $(\mathcal{P}, \mathcal{C}, \models)$, where $\mathcal{C}$ is a partially ordered set, and where $\models \subseteq \mathcal{P} \times \mathcal{T} \times \mathcal{C}$ is a relation satisfying*

$$\forall_{p\in\mathcal{P}}\forall_{A,B\in\mathcal{C}}\forall_{t\in\mathcal{T}} \quad A \leq B \wedge p \models_t A \Rightarrow p \models_t B \ . \quad (1)$$

The elements of $\mathcal{C}$ are called constraints.

The intuitive meaning of the statement $p \models_t C$ is that the principal p satisfies constraint C at time t. Requirement (1) is just expressing the fact that if $A \leq B$, then A is a more restrictive constraint than B, independent of the time t. The intention is that $p \models_t C$ could be considered as a "stochastic process" with boolean values; at each point in time the constraint C ("randomly") defines a subset of $\mathcal{P}$ satisfying (1).

Group membership conditions is not the only type of constraints possible. Besides depending on the principal and the time, constraints could depend on e.g. local and global data, and the security context in which the principal is acting. Thus, besides group membership conditions, typical examples include role occupancy, time, and conditions on values in different fields in some external database. One could also allow constraints to contain side effects like e.g. audit labelling, incrementing of counters, etc.

Constraints, now, are put together in strings, *chain constraints*, to form the basic mechanism for transfer of authorisation, as described in the introduction. Initially we do not commit further to a specific notation for sets of chain constraints, and consider just arbitrary languages. We return to the issue of notations in section 4.

Definition 2 (chain constraint) *Given a constraint structure $(\mathcal{P}, \mathcal{C}, \models)$, the set of* chain constraints, *$\mathcal{C}^*$ (Kleene star), is defined as the set of all strings over the alphabet $\mathcal{C}$. $\mathcal{C}^*$ is a partially ordered set. If $\sigma = A_1 A_2 \ldots A_m$ and $\tau = B_1 B_2 \ldots B_n$ are chain constraints, then $\sigma \leq \tau$ if and only if $n = m$ and $A_i \leq B_i$ for $i = 1, \ldots m$.*

The empty string is denoted by ε, and the length of $\sigma \in \mathcal{C}^*$ is $|\sigma|$. Observe that, according to def. 2, the only chain constraint that is greater or equal, or less or equal, to ε, is ε itself.

A chain constraint is a way of describing restrictions on a delegation chain. E.g., the chain constraint $\sigma = ABBC \in \mathcal{C}^*$ allows delegation chains of length four ($= |\sigma|$) which begin with a principal satisfying A, are continued by two principals (one after the other) satisfying B and end with a principal satisfying C. Such a delegation chain is said to *satisfy* the chain constraint σ. When the notion of a *delegation chain* is properly defined in section 3, it will follow immediately from definition 1 that if a delegation chain satisfies σ and $\sigma \leq \tau$, then the delegation chain also satisfies τ.

Chain constraints are used to control delegations of authorisations. In this paper, a set of *authorisations* $\mathcal{A}$ is an (abstract) partially ordered set. If $\alpha, \beta \in \mathcal{A}$ and $\alpha \leq \beta$, then the interpretation is that the authorisation β entails the authorisation α, i.e. if a principal has authorisation β, the principal also has authorisation α. This will be made precise in the definition of the *authorisation relation*.

Definition 3 (constraint certificate) *A* constraint certificate, *or just* certificate *for short, is a four-tuple $d = (L, \alpha, t, p)$ where $L \subseteq \mathcal{C}^*, \alpha \in \mathcal{A}, t \in \mathcal{T} \cup \{-1\}$, and $p \in \mathcal{P}$. We normally write such a d as $(L, \alpha, t)_p$, and say that d is* signed *or* issued *by the principal p. The number t is called the* time–stamp *of the certificate.*

The intended meaning of a certificate $(L, \alpha, t)_p$ is the following: at time t the principal p is signing a statement permitting delegation of the authorisation α, provided that

the resulting delegation chain satisfies the different constraints in some chain constraint in L at the future points in time when the respective delegation steps are made. How this is done is made precise in the following subsection.

2.2 The Certificate Database

A *certificate database* $\mathcal{D}$ is a finite set of certificates that changes over time. The set $\mathcal{D}_t$ contains the certificates of the database at time t and is referred to as the *state* of the database at time t. It is required that t is strictly greater than all the time–stamps of the certificates contained in $\mathcal{D}_t$ (this will automatically follow from the state change definitions). To avoid trivialities, the database is assumed to be non–empty at time $t = 0$. All the certificates in $\mathcal{D}_0$ have time–stamp $t = -1$ and are called *initial* certificates of $\mathcal{D}$.

The idea is that given a certificate database at some point in time, a principal may request a state change. A decision is made, on the basis of the information in the current state, whether the request is granted or not. If the request is granted, the database is updated accordingly.

Definition 4 (state change: declare) *Given a database with state $\mathcal{D}_t$, let $d = (L, \alpha, t)_p$ be a certificate. The state change* declare

$$\mathcal{D}_t \xrightarrow{\text{declare} : d} \mathcal{D}_{t+1}$$

is defined as follows: if there exists a certificate $d' = (L', \alpha', t')_{p'} \in \mathcal{D}_t$ such that $\alpha \leq \alpha'$ and

$$\forall_{\omega \in L} \exists_{A' \in \mathcal{C}} \exists_{\omega' \in \mathcal{C}^*} \quad \omega \leq \omega' \ \wedge \ A'\omega' \in L' \ \wedge \ p \models_t A' \ , \tag{2}$$

then the certificate d is accepted and $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{d\}$, otherwise $\mathcal{D}_{t+1} = \mathcal{D}_t$. If d is accepted, we say that d is derivable *from d' (note that there could be several such d':s).*

Let us instead consider the question: what certificates can p declare at time t such that they can be derived from d'? Since p must obey the conditions given in d', p's authorisation α is bounded by the authorisation α' given in d'. Furthermore, p (and p's set of chain constraints L) must satisfy the set of chain constraints L' given in d'. This amounts to the following (which is contained in condition (2)):

1. First extract all chain constraints from L' having as its first symbol a constraint that is satisfied by p at time t, i.e. let

$$L_1 = \{A'\omega' \in L' \mid p \models_t A'\} \ .$$

2. Then delete the first symbol (the one corresponding to p) from all strings in L_1, i.e. let

$$L_2 = \{\omega' \mid \exists_{A' \in \mathcal{C}} A'\omega' \in L_1\} \ .$$

This set is the weakest set of chain constraints p can use for the delegation, or, put in another way, $(L_2, \alpha', t)_p$ is the most powerful delegation p can derive from d'.

3. p can now choose to restrict L_2 to any subset $L_3 \subseteq L_2$.

4. Finally, p can choose to restrict any of the chain constraints $\omega' \in L_3$ to $\omega \leq \omega'$, thus obtaining L_4, a valid set of chain constraints for p's delegation.

This process can be described in two steps: first extract the set L_2 from L' and then restrict L_2 to the set L_4. To capture these two steps we introduce two notations. We begin with the restriction by defining a preorder on $2^{\mathcal{C}^*}$. If $M, N \subseteq \mathcal{C}^*$, then $M \leq N$ if and only if

$$\forall_{\omega_1 \in M} \exists_{w_2 \in N} \quad w_1 \leq w_2 \ .$$

Next we define the *extraction operator* $\mathrm{E} : \mathcal{P} \times 2^{\mathcal{C}^*} \times \mathcal{T} \to 2^{\mathcal{C}^*}$ as

$$\mathrm{E}\,(p, M, t) = \{\omega \mid \exists_{A \in \mathcal{C}} A\omega \in M \ \wedge \ p \models_t A\} \ .$$

Its clear that $L_2 = \mathrm{E}\,(p, L', t)$ and $L_4 \leq L_2$ in the process description above (items 1-4). We can now rephrase definition 4 in a more compact form. The certificate $d = (L, \alpha, t)_p$ is accepted (at time t) if and only if there exists a certificate $d' = (L', \alpha', t')_{p'} \in \mathcal{D}_t$ such that $\alpha \leq \alpha'$ and $L \leq \mathrm{E}\,(p, L', t)$.

Example 1 Assume that p delegates the authorisation α at time t, using the set of chain constraints

$$L = \{\, A_1A_2A_3,\ B_1B_2B_3B_4,\ C_1C_2,\ D_1D_2D_3 \,\} \ ,$$

by declaring the certificate $d = (L, \alpha, t)_p$.

Now, suppose that q decides to delegate the authorisation $\alpha' \leq \alpha$ one step further at time $t' > t$. Say that q satisfies A_1, C_1 and D_1, but not B_1, at time t'. Then

$$\mathrm{E}\,(q, L, t') = \{\, A_2A_3,\ C_2,\ D_2D_3 \,\}$$

is the weakest set of chain constraints q can derive from d at time t'. If A_2', A_3' and C_2' are constraints satisfying $A_2' \leq A_2$, $A_3' \leq A_3$ and $C_2' \leq C_2$, then q could e.g. choose to restrict $\mathrm{E}\,(q, L, t')$ to

$$L' = \{\, A_2'A_3',\ C_2' \,\} \leq \mathrm{E}\,(q, L, t') \ .$$

Finally, q (successfully) declares the certificate $d' = (L', \alpha', t')_q$.

We have defined how delegation of an authorisation takes place. Now we define the result of a delegation chain, i.e. which principals are possible receivers of the authorisation.

Definition 5 (authorisation relation) *Given a certificate database $\mathcal{D}$ with constraint structure $(\mathcal{P}, \mathcal{C}, \models)$ and authorisation set $\mathcal{A}$, we define the* authorisation relation $\mathrm{Auth} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{T}$ *as follows:*

$\mathrm{Auth}\,(p, \alpha, t)$ *is true if and only if there exists a certificate $d' = (L', \alpha', t')_{p'}$ in $\mathcal{D}_t$ and a chain constraint of length one $A \in L'$ such that*

$$p \models_t A \wedge \alpha \leq \alpha' \ . \tag{3}$$

In this case we say that p's authorisation α is derivable *from d' at time t.*

The authorisation relation answers the question: does the principal p have the authorisation α at time t. The first condition of (3) ensures that p is permitted as the last principal in a delegation chain at time t. The second condition ensures that the requested authorisation α is entailed by the authorisation α' given in the used certificate.

Note that the last constraint in a chain constraint corresponds to the principal requesting the authorisation, not the principal declaring the last delegation step. Also note that, using the extraction operator, definition 5 could be expressed as: $\mathrm{Auth}\,(p, \alpha, t)$ is true if and only if there exists a certificate $d' = (L', \alpha', t')_{p'}$ in $\mathcal{D}_t$ such that

$$\varepsilon \in \mathrm{E}\,(p, L', t) \wedge \alpha \leq \alpha' \ .$$

In example 1 above, any principal r satisfying C_2' at time $t'' > t'$ could enjoy the authorisation α', since $\varepsilon \in \mathrm{E}\,(r, L', t'')$ if and only if $r \models_{t''} C_2'$.

3 Soundness

To prove soundness of the authorisation relation (and to make the semantics of the certificates precise) we need to formalise the concept of a delegation chain. A *delegation chain* of length n is a list

$$\mu = [(p_1, \alpha_1, t_1), (p_2, \alpha_2, t_2), \ldots, (p_n, \alpha_n, t_n)] \ ,$$

where $p_1, \ldots, p_n \in \mathcal{P}$, $\alpha_1, \ldots, \alpha_n \in \mathcal{A}$ and $t_1, \ldots, t_n \in \mathcal{T}$. The interpretation of μ is that p_i delegates authority α_i at time t_i to p_{i+1} for $i = 1, \ldots, n-1$, and that p_n *has* authorisation α_n at time t_n.

Given a certificate $d = (L, \alpha, t)_p$, we say that the delegation chain μ *satisfies* the certificate d if all of the following conditions hold:

1. $t < t_1 < t_2 < \ldots < t_n$
2. $\alpha \geq \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n$
3. There exist $A_1, \ldots, A_n \in \mathcal{C}$ such that $A_1 \ldots A_n \in L$ and $p_i \models_{t_i} A_i$ for $i = 1, \ldots n$.

If μ satisfies d, then μ was one of the delegation chains p intended to permit, since (1) assures that the certificates have arrived in the correct order, (2) assures that no authorisation originating from p's certificate is more powerful than α, and (3) assures that there is a chain constraint of length n in L such that each constraint in this chain constraint is satisfied by the appropriate principal at the relevant time.

Now we are in a position to prove a soundness result, soundness in the sense that if a principal receives an authorisation, each sub–chain[1] of the entire delegation chain satisfies a corresponding certificate. First we need a lemma.

Lemma 1 *Assume the following*

- $\mathrm{Auth}\,(p_n, \alpha_n, t_n)$.
- $d = d_0 = (L, \alpha, t)_p \in \mathcal{D}_{t_1}$.
- *For all* $i : 1 \leq i < n$, $d_i = (L_i, \alpha_i, t_i)_{p_i} \in \mathcal{D}_{t_{i+1}}$.
- $\mathrm{Auth}\,(p_n, \alpha_n, t_n)$ *is derivable from* d_{n-1}.
- *For all* $i : 1 \leq i < n$, d_i *is derivable from* d_{i-1}.

Then the delegation chain

$$\mu = [(p_1, \alpha_1, t_1), (p_2, \alpha_2, t_2), \ldots, (p_n, \alpha_n, t_n)]$$

satisfies the certificate d.

Proof. Since $\mathrm{Auth}\,(p_n, \alpha_n, t_n)$ was derived from d_{n-1}, d_{n-1} was derived from d_{n-2} and so on until d, it follows immediately from definition 4 and definition 5 that $t < t_1 < t_2 < \ldots < t_n$ and $\alpha \geq \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n$.

Before we prove the last part, we prove the following claim. For any $\omega' \in \mathcal{C}^*$, $p' \in \mathcal{P}$, $L' \subseteq \mathcal{C}^*$ and $t' \in \mathcal{T}$,

$$\{\omega'\} \leq \mathrm{E}\,(p', L', t') \Rightarrow \exists_{A' \in \mathcal{C}}\ \{A'\omega'\} \leq L' \wedge p' \models_{t'} A' \ . \tag{4}$$

If $\{\omega'\} \leq \mathrm{E}\,(p', L', t')$, then there exists $\omega'' \in \mathrm{E}\,(p', L', t')$ such that $\omega' \leq \omega''$ by the definition of the preorder. Furthermore, by the definition of $\mathrm{E}\,(p', L', t')$, there exists $A' \in \mathcal{C}$ such that $A'\omega'' \in L'$ and $p' \models_{t'} A'$. Since $\omega' \leq \omega''$ we get $A'\omega' \leq A'\omega''$, and hence $\{A'\omega'\} \leq L'$. Thus, the claim is proved.

To streamline the argument, let $d_0 = d$ and $L_0 = L$. We prove the following by induction from $k = n$ down to $k = 1$:

There exist $A_k, A_{k+1}, \ldots, A_n \in \mathcal{C}$ such that $\{A_k A_{k+1} \ldots A_n\} \leq L_{k-1}$ and $p_i \models_{t_i} A_i$ for $i = k, k+1, \ldots, n$.

The assumption that $\mathrm{Auth}\,(p_n, \alpha_n, t_n)$ was derived from d_{n-1} implies that $\{\varepsilon\} \leq \mathrm{E}\,(p_n, L_{n-1}, t_n)$. By (4), there exists $A_n \in \mathcal{C}$ such that $\{A_n\} \leq L_{n-1}$ and $p_n \models_{t_n} A_n$.

[1] A *sub–chain* of a delegation chain μ is a delegation chain obtained by deleting an initial segment of μ.

Thus, the case $k = n$ is proved. Assuming the induction hypothesis for some k, $2 \leq k \leq n$, and using the fact that d_{k-1} was derived from d_{k-2}, we find that

$$\{A_k A_{k+1} \dots A_n\} \leq L_{k-1} \leq \mathrm{E}\,(p_{k-1}, L_{k-2}, t_{k-1}) \quad .$$

Again, by (4), there exists $A_{k-1} \in \mathcal{C}$ such that $\{A_{k-1} A_k \dots A_n\} \leq L_{k-2}$ and $p_{k-1} \models_{t_{k-1}} A_{k-1}$, proving the induction step.

We have thus proved that there exist $A_1, A_2, \dots, A_n \in \mathcal{C}$ such that

$$\{A_1 A_2 \dots A_n\} \leq L_0 = L$$

and

$$p_i \models_{t_i} A_i \text{ for } i = 1, \dots, n \quad .$$

Since $\{A_1 A_2 \dots A_n\} \leq L_0 = L$, there exist $B_1, B_2, \dots, B_n \in \mathcal{C}$ such that $B_1 B_2 \dots B_n \in L$ and $A_i \leq B_i$ for $i = 1, \dots, n$. The definition of $\models$ implies that $p_i \models_{t_i} B_i$ for $i = 1, \dots, n$, proving that the delegation chain μ satisfies the certificate d. □

Using this lemma we now prove the following soundness result.

Theorem 1 (soundness) *Assume that* $\mathrm{Auth}\,(p, \alpha, t)$. *Then there exists a sequence of certificates*

$$d_0 = (L_0, \alpha_0, -1)_{p_0} \in \mathcal{D}_0, \quad d_i = (L_i, \alpha_i, t_i)_{p_i} \in \mathcal{D}_{t_{i+1}}$$

for $i = 1, \dots, n-1$ *, where* $t_n = t$, *and a delegation chain*

$$\mu = [(p_1, \alpha_1, t_1), \dots, (p_{n-1}, \alpha_{n-1}, t_{n-1}), (p, \alpha, t)]$$

such that each sub–chain

$$\mu_i = [(p_{i+1}, \alpha_{i+1}, t_{i+1}), \dots, (p, \alpha, t)]$$

satisfies the corresponding certificate d_i.

Proof. It's immediate from definition 5 that if $\mathrm{Auth}\,(p, \alpha, t)$ holds, then this authorisation can be derived from some certificate $d' \in \mathcal{D}_t$ with time–stamp t'. If $t' \neq -1$, then d' was added to the certificate database by the state change *declare* (cf. def. 4), and hence there exists a certificate $d'' \in \mathcal{D}_{t'}$ with time–stamp t'' from which d' can be derived. Again, either $t'' = -1$ or d'' can be derived from some $d''' \in \mathcal{D}_{t''}$.

This process must terminate since the time–stamps of the certificates form a strictly decreasing sequence of integers bounded from below by -1. Assume that this process halts after n steps. This means that we have reached a certificate with time–stamp -1. If we index the certificates (and their contents) as above, we get the sequence of certificates $d_0, \dots, d_{n-1}$ such that the d_i can be derived from the d_{i-1}, and where $\mathrm{Auth}\,(p, \alpha, t)$ can be derived from d_{n-1}.

The theorem now follows by repeatedly applying lemma 1 to each sub–chain μ_i, $0 \leq i \leq n-1$. □

Note that *completeness* in this context means that if a sequence of linked certificates, starting with an initial certificate, has been declared and all sub–chains of a certain delegation chain satisfy the corresponding certificates, then the last tuple in the delegation chain should belong to the relation Auth. But this is immediate since the assumption that all sub–chains satisfy the corresponding certificates also implies that the shortest sub–chain (the one of length 1) satisfies the last certificate in the certificate chain and this is precisely the definition of the Auth relation. On the other hand, it is not meaningful to exclude the chain of length one, since then the principal declaring the last certificate can always make sure that a particular principal will not receive the authorisation in question.

4 Regular Chain Constraints

We now turn to the issue of identifying a suitable representation for sets of chain constraints. There is considerable scope for variability. The trade-off, as ever, is between simplicity of expression, algorithmic tractability, and application needs. The obvious first choice is some suitable fragment of regular expressions. Richer languages can be considered too. However, as yet we have found no real use for expressive power going beyond that of the regular languages. In fact, the suggestion we make in this section is for a very simple language which just barely generalises ACL's to include a restricted form of Kleene star. Let us say that a *simple regular expression* (that defines a *simple regular language*) over the alphabet $\mathcal{C}$ is an expression of the form: $\omega = A_1^{k_1} A_2^{k_2} \dots A_n^{k_n}$, where $A_i \in \mathcal{C}$ and $k_i \in \{1, *\}$ for $i = 1, \dots, n$. $\mathcal{L}(\omega) \subseteq \mathcal{C}^*$ will denote the language it represents. A simple regular expression ω is said to be *initially fixed* if $k_1 = 1$; this implies that all strings in $\mathcal{L}(\omega)$ begin with the same symbol (A_1 in the notation above). Nothing in the framework forces to adopt this requirement. However, we find it reasonable to require that certificates identify explicitly and uniquely the initial constraint/receiver of delegation.

When restricted to initially fixed simple regular expressions, the extraction operator becomes very simple to compute:

$$\mathrm{E}\left(p, \mathcal{L}(A_1 A_2^{k_2} \dots A_n^{k_n}), t\right) = \begin{cases} \mathcal{L}(A_2^{k_2} \dots A_n^{k_n}) & \text{if } p \models_t A_1, \\ \emptyset & \text{otherwise} \end{cases}$$

and the result (if not empty) becomes a simple regular language. To retain this property inductively, we require that the principal restricting this set of chain constraints, restricts it to an *initially fixed* simple regular language.

The rationale for this requirement is (besides to keep things uniform and simple) that certificates of the form

$d = (\mathcal{L}(A^*\omega), \alpha, t)_p$ are essentially superfluous, since any delegation chain satisfying d has a sub–chain satisfying d *and* satisfying $d' = (\mathcal{L}(\omega), \alpha, t)_p$. In particular, if d was accepted at time t, then d' would also have been accepted at time t. Providing A with the right to delegate some authority to ω seems a bit pointless, if at the same time ω receives that authority directly.

The only exception to this argument is if $\varepsilon \in \mathcal{L}(\omega)$ and the sub–chain referred to above is empty. In this case ω is of the form $\omega = A_1^* A_2^* \dots A_n^*$ (all exponents are Kleene stars), including the case $\omega = \varepsilon$. We could mimic the argument above by replacing d' above with $d' = (\mathcal{L}(A \cup A_1 \cup \dots \cup A_n), \alpha, t)_p$. The reason the argument works, in this case, is that any delegation chain (of length ≥ 1) satisfying $d = (\mathcal{L}(A^* A_1^* A_2^* \dots A_n^*), \alpha, t)_p$ has a sub–chain of *length one* satisfying d and d'. Furthermore, as above, if d was accepted at time t, then d' would also have been accepted at time t.

The problem is that $A \cup A_1 \cup \dots \cup A_n$ is not a *simple* regular expression. This could easily be solved in practice by, instead of declaring one certificate with chain constraint set $\mathcal{L}(A^* A_1^* A_2^* \dots A_n^*)$, declaring $n+1$ certificates having chain constraint sets $\mathcal{L}(A_1), \dots, \mathcal{L}(A_n)$ and $\mathcal{L}(A)$, respectively. Formally though, these new certificates might not all be accepted instead of d, since we only allow *one* certificate to be declared at any single point in time. A slight modification to the state change 'declare', namely to allow a *set* of (independent) certificates to be declared simultaneously, or alternatively permit unions of initially fixed regular expressions (see below), would solve this little problem.

Now, given a certificate

$$d = \left(\mathcal{L}(A A_1^{k_1} \dots A_n^{k_n}), \alpha, t_0\right)_q,$$

and assuming that $p \models_t A$ (where $t > t_0$), what sets of chain constraints L can p use when declaring a certificate $d' = (L, \alpha', t)_q$ (where $\alpha' \leq \alpha$), given that d' should be derivable from d?

Since we now are restricting ourselves to initially fixed simple regular expressions, any such regular language L bounded from above by $\mathcal{L}(A_1^{k_1}, \dots, A_n^{k_n})$ is permitted as chain constraint set for the certificate d'. This implies that L has the form $L = \mathcal{L}(\omega_1 \dots \omega_n)$ where $\omega_1, \dots, \omega_n$ are simple regular expressions, $\omega_1 \dots \omega_n$ is initially fixed, and where ω_i has one of the following two forms:

1. If $k_i = 1$, then $\omega_i = B_i$ for some $B_i \in \mathcal{C}$ which satisfies $B_i \leq A_i$.

2. If $k_i = *$, then $\omega_i = B_{i1}^{l_{i1}} \dots B_{in_i}^{l_{in_i}}$ ($n_i \geq 0$) for some $B_{i1}, \dots, B_{in_i} \in \mathcal{C}$ which all satisfy $B_{ij} \leq A_i$ and where $l_{ij} \in \{1, *\}$. Note that $\omega_i = \varepsilon$ if $n_i = 0$.

Typical examples of useful chain constraint sets include $\mathcal{L}(AB^*)$, $\mathcal{L}(ABB^*)$, $\mathcal{L}(AB^*C)$, $\mathcal{L}(ABB^*C)$ and $\mathcal{L}(AB^*CD^*)$. If we, for example, assume that chain constraints only represent group membership, then the (informal) semantics of these sets could roughly be described as follows:

$\mathcal{L}(AB^*)$ Any member of A can be the root of a "management tree", managing the authorisation (by delegation) within the the group B, *and* members of A can enjoy the authorisation themselves.

$\mathcal{L}(ABB^*)$ Same as the previous item, except that the members of A do not receive the authorisation themselves (assuming that $A \cap B = \emptyset$).

$\mathcal{L}(AB^*C)$ Any member of A can delegate the right to members of B to create a management structure (within B) for managing the authorisation of members in C, *and* members of A are also permitted to authorise members of C directly. Assuming that $B \cap C = \emptyset$, this (and also the next item) exemplifies "outsourcing"; the administration of the authorisation within the group C is handled by B, including the right to organise the work within B as they see fit.

$\mathcal{L}(ABB^*C)$ Same as the previous item, except that the members of A are not permitted to bypass the administrator group B.

$\mathcal{L}(AB^*CD^*)$ In this example, B and D may be groups in two different organisations. In this case it may be desirable to constrain A's delegational powers so that any administrative structures leading from A to D must pass by some particular group, C, of key account managers, or liaison officers.

We finish this section with a larger example and some possible extensions. The example will exemplify how regular chain constraint sets can be used to gradually establish management structures for managing authorisations.

Example 2 To simplify the notation in this example we will use the names of principals and organisations (= groups) as constraints with the obvious meaning. Lower–case letters denote principals and upper–case letters denote organisations. Principals represented by lower–case letters belong to the corresponding upper–case organisation. We will assume that the organisations B, C, E, and F all are contained in A, and that the organisation D is contained in C.

Figure 2 depicts a possible delegation tree resulting from the certificate $(pA^*, \alpha, t_0)_r$. The labels of the nodes in the tree represent the principals who are delegating and/or receiving authorisations. The labels of the edges in the tree represent the regular chain constraint sets used in declared

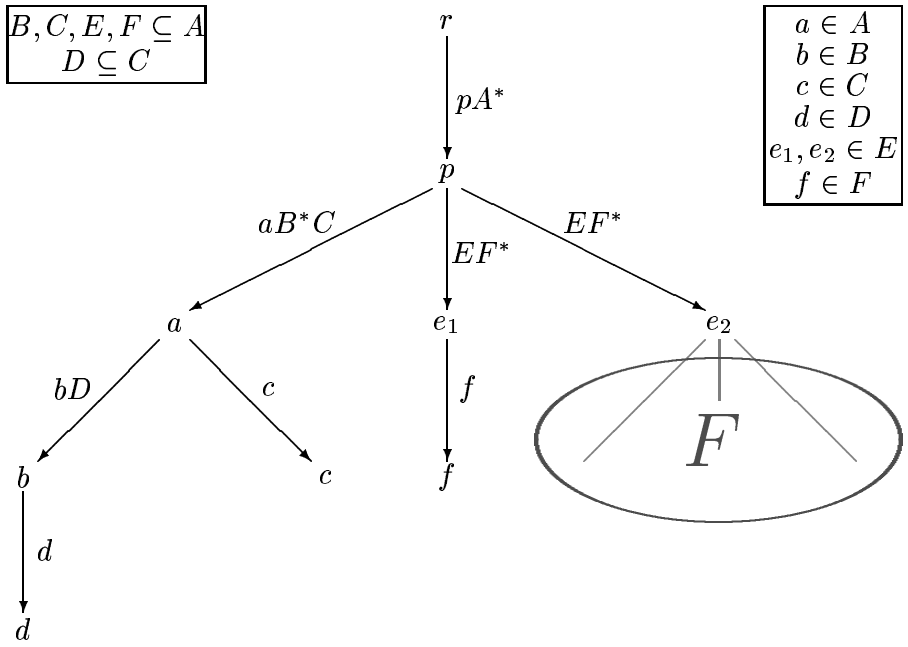

Figure 2. A delegation tree

certificates. Outgoing edges correspond to certificates declared, and incoming edges correspond to delegation powers/authorisations received. Furthermore, in each delegation step, the authorisation α is (possibly) restricted further (not shown in the figure).

Now, let us examine a few steps in this delegation tree. In the second step of the delegation chain, p satisfies the leftmost constraint in pA^*, and hence, can extract A^*. The principal p chooses to restrict A^* to aB^*C and EF^* in the two certificates p declares. These restrictions are permitted since $\mathcal{L}(aB^*C) \leq \mathcal{L}(A^*)$ and $\mathcal{L}(EF^*) \leq \mathcal{L}(A^*)$.

The principals e_1 and e_2 both satisfy the constraint E, and can therefore successfully extract F^* from EF^*. In e_1's certificate, F^* is restricted to f (permitted, since $\mathcal{L}(f) \leq \mathcal{L}(F^*)$), thereby authorising the principal f. The principal e_2, on the other hand, decides to build a larger subtree (which necessarily lives entirely within the organisation F) by restricting F^* in some suitable fashion, and so on.

The choice of initially fixed simple regular expressions is somewhat arbitrary. The framework presented in this paper clearly supports more general sets of chain constraints. Obvious extensions might include *unions* of initially fixed simple regular expressions and/or notation that enables implicit unions. One could e.g. allow expressions of the type $A_1 A_2^{k_2} \ldots A_n^{k_n}$, where $k_i \in \{1, *\}$ *or* $k_i = [i, j]$ for non negative integers $i < j$. The expression $A^{[i,j]}$ is then interpreted as a shorthand for the regular expression $A^i \cup A^{i+1} \cup \ldots \cup A^j$. This would allow a principal to flexibly express restrictions on the length of (parts of) delegation chains in a single certificate (note that this can in principle be achieved by declaring several certificates).

5 Scenario: Collaborating Organisations

In this section we introduce a specific scenario in order to illustrate some of the capabilities of constrained delegation, and the ways they could be used in practice. The scenario is based on the case of a number of national task forces delegating authority to a common UN high command (*UNHC*).

Each national task force will have a National Task Force Command (*NTFC*) associated with it. For the sake of the example, let *NTFC*(S) be the *NTFC* belonging to Sweden. The *NTFC* will be the "owner" of each of the national task forces in the sense that it will from the outset possess all administrative privileges concerning that entity. In particular, *NTFC*(S) may have assigned to it free delegational powers in terms of a certificate

$$c_0 = (\textit{NTFC}(S)\ \textit{any}^*, \textit{NTFC}(S)\text{-}\textit{resources}, t_0)_p$$

where, most likely, p is the Swedish National High Command. The first component of c_0 is the regular expression delineating the possible delegation chains (in this case any such chain must originate in the Swedish National Task Force Command, and they in turn are empowered to delegate authority as they see fit). The second component,

NTFC(S)*-resources*, indicates the scope of the delegatable authority in this case, and t_0 is the certificate time stamp. So in this case, the certificate is intended to empower *NTFC*(S) to delegate in any way it sees fit, any authority concerning its own resources.

First we consider the case where *NTFC*(S), using certificate c_0, delegates to *UNHC* the authority to delegate, in any number of steps, using UN-affiliated personnel, read access to some Swedish surveillance information. The corresponding certificate can have the following shape:

$$c_1 = (\textit{UNHC UNHC-stf}^* \textit{ UN-stf}, \textit{s-info}(S), t_1)_{\textit{NTFC}_{(S)}}.$$

The UN High Command can now use c_1 to provide UN High Command staff (which will be a larger group than *UNHC*) with administrative and decentralisable power to provide UN-affiliated staff with access to Swedish surveillance information. For instance, in the following certificate, UN High Command staff has received, from UN High Command, administrative rights to provide operative UN staff of some nationality, say C, with access to Swedish surveillance information:

$$c_2 = (\textit{UNHC-stf op}(C), \textit{s-info}(S), t_2)_{\textit{UNHC}}$$

UN High Command staff can then use this certificate to support the following authorisation

$$c_3 = (\textit{spec-op}(C), \textit{spec-s-info}(S), t_3)_{\textit{UNHC-stf}}$$

In the process of issuing c_3, UN High Command staff has constrained the scope of operative personnel and access rights in relation to the certificate c_2. Observe that operative personnel of nationality C will not by this certificate receive any delegational powers regarding Swedish surveillance information.

The second example is intended to illustrate the power and flexibility obtained when constraints are generalised to cover not only group affiliation properties, but also more general constraints related e.g. to time or the holding of some condition. The intention is that *NTFC*(C) might want to authorise *UNHC* to, in an emergency situation, through administrative channels set up by UN High Command, give Swedish operative forces some privileges concerning supplies belonging to C. We use a tuple-like notation for conjunction of constraints, so that e.g. $(\textit{UNHC-stf}, \textit{alert})$ represents the conjunction of constraints that the issuing principal belongs to the group *UNHC-stf* and that the condition *alert* holds. In this way, *NTFC*(C) might issue the certificate

$$(\textit{UNHC UN-stf}^* \ (\textit{UN-stf}, \textit{alert}) \ \textit{op}(S), \textit{spl}(C), t)_{\textit{NTFC}_{(C)}}$$

empowering *UNHC* to set up an administrative organisation at will for administering access by Swedish operative forces to C's supplies, but preventing the right to access to be ultimately granted until an emergency condition holds. Many variations on such a scheme are possible, including threshold-like ones where several specific parties must have taken part in a delegation chain for the operative authority to be possible to take effect.

6 Conclusion

We have argued that some applications, with outsourcing as an archetypical example, would benefit from a more fine-grained and flexible control over delegation than current models admit. The standard approach to delegation is binary: Either delegation is possible, and then no substantial further control over the way it is used is possible, or else no delegation is permitted. Some authors (cf. [8]) go beyond this by permitting a fixed upper bound to be imposed on the depth of delegation chains. We have introduced a model which permits much finer control over the scope of delegations. The central idea is to introduce (regular) expressions that constrain the possible shapes of delegation chains, for aspects such as depth, group/role memberships, timing constraints, other constraints depending on the current security context, or just constraints depending on external data. In this way it becomes possible to define administrative structures in a more gradual and uniform way.

Our purpose with this paper has been to introduce and motivate the basic model. We have not, for instance, touched upon the issue of revocation. One set of problems pertaining to the handling of dependencies arise in the context of certificate chaining (cf. [6, 4] for recent work in this direction). Other issues arise once one starts to admit revocation as a delegatable action: Who should be permitted to revoke a given certificate, and how should this be reflected in the delegation logic? Concerning distribution of revocation information we did not find particular challenges which are not found equally in other related work, and so we view this as somewhat orthogonal to the issues discussed here.

Another set of issues which we have not addressed concerns the choice, design, and implementation of computational models to support constrained delegation. Our intention has been to keep the basic model as free of bias towards any particular implementation regime as possible. In principle the constrained delegation model can be applied to a wide variety of representation and storage architectures (say: ACL's, directories, attribute certificates, centralised or decentralised storage models), as well as enforcement models (push, pull, or combinations). Key functionality which will be reported in a forthcoming paper is the efficient representation and resolution of constraints, and the management of delegation chains.

Acknowledgements Thanks are due to Dieter Gollmann of Microsoft Research, Cambridge, and to Andres Martinelli,

KTH, for many discussions on this and related topics.

References

[1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The Role of Trust Management in Distributed Systems Security. In Vitek and Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*. Springer-Verlag, 1999.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralised Trust Management. In *Proceedings of the 17th Symposium on Security and Privacy*, pages 164 – 173, Los Alamitos, 1996. IEEE Computer Society Press.

[3] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. SPKI Certificate Theory, May 1999. Published online: http://www.ietf.org/internet-drafts/draft-ietf-spki-cert-theory-0.5.txt.

[4] B. S. Firozabadi and M. Sergot. Revocation Schemes for Delegated Authorities. In *Proceeding of Policy 2002: IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*. IEEE, June 2002. In press.

[5] B. S. Firozabadi, M. Sergot, and O. Bandmann. Using Authority Certificates to Create Management Structures. In *Proceeding of Security Protocols, 9th International Workshop*, Cambridge, UK, April 2001. Springer Verlag. In press.

[6] Å. Hagström, S. Jajodia, F. Parisi.Persicce, and D. Wijesekera. Revocation - a Classification. In *The Proceeding of the 14th Computer Security Foundation Workshop*. IEEE press, 2001.

[7] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: Assigning roles to strangers. In *IEEE Symposium on Security and Privacy*, pages 2–14, 2000.

[8] Li, Grosof, and Feigenbaum. A Logic-based Knowledge Representation for Authorization with Delegation. In *PCSFW: Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.